User Manual

3-Heights™ PDF Merge Split API

Version 4.5

PDF-TOOLS.COM
Premium PDF Technology

Contents

1 Introduction . . . 1

2 Installation . . . 1

2.1 Windows . . . 1

2.2 Unix . . . 1

2.3 Uninstall, Install a New Version . . . 3

3 License Management . . . 4

3.1 Graphical License Manager Tool . . . 4

3.2 Command Line License Manager Tool . . . 5

3.3 License Key Storage . . . 5

4 Programming Interfaces . . . 6

4.1 Visual Basic 6 . . . 6

4.2 .NET . . . 7

5 User's Guide . . . 9

5.1 Basics . . . 9

5.2 How to Create PDF/A Compliant Documents . . . 9

5.3 How to Efficiently Use the API . . . 10

6 Programmer's Reference . . . 11

6.1 The InDoc Interface . . . 11

Close . . . 11
CropBox . . . 11
ErrorCode . . . 11
ErrorMessage . . . 12
GetXMPMetadataMem . . . 12
GetXMPMetadata . . . 12
InfoEntry . . . 12
MediaBox . . . 12
Open . . . 12
OpenMem . . . 13
Page . . . 13
PageCount . . . 13
Rotate . . . 13

6.2 The OutDoc Interface . . . 14

AddAssociatedFile . . . 14
AddEmbeddedFile . . . 14
AddOutlineItem . . . 14
AddOutlineItem2 . . . 15
Author . . . 16
Close . . . 16
CopyAssociatedFiles . . . 16
CopyEmbeddedFiles . . . 16
CopyForms . . . 17
CopyLogicalStructure . . . 17
CopyMetadata . . . 17
CopyOptionalContent . . . 17
CopyOutlines . . . 17
CopyOutlineItems . . . 17
CopyOutputIntent . . . 18
CopyPages . . . 18
CopyViewerProperties . . . 18
Create . . . 18
CreateInMemory . . . 19
ErrorCode . . . 20
ErrorMessage . . . 20
FlattenFormFields . . . 20
FlattenSigAppearance . . . 20
GetPDF . . . 20
InfoEntry . . . 21
Keywords . . . 21
LicenseIsValid . . . 21

MergeOptionalContent . . . 21
OptimizeResources . . . 21
OutputIntent . . . 22
PageLayout . . . 22
PageMode . . . 22
RemoveNamedDests . . . 22
Rotate . . . 22
SetLicenseKey . . . 22
SetOpenAction . . . 22
SetViewerPreference . . . 23
SetXMPMetadata . . . 23
SetXMPMetadataMem . . . 23
Subject . . . 23
Title . . . 23
6.3 Enumerations . . . 24
TPDFErrorCode . . . 24
TPDFPermission . . . 24
7 Examples . . . 25
8 Licensing and Copyright . . . 25
9 Contact . . . 26

1 Introduction

The purpose of the 3-Heights™ PDF Merge Split API is to read PDF files and rearrange any of the pages in an arbitrary order. The component provides COM interfaces that can be used in any programming language that supports the COM technology such as C++, Visual Basic, ASP and many more.

2 Installation

2.1 Windows

The retail version of the 3-Heights™ PDF Merge Split API comes as a ZIP archive containing various files including runtime binary executable code, files required for the developer, documentation and license terms.

1. Download the ZIP archive of the product from your download account at http://www.pdf-tools.com.
2. Unzip the file using a tool like WinZip available from WinZip Computing, Inc. at http://www.winzip.com to a directory on your hard disk where your program files reside (e.g. *C:\Program Files\PDF Tools AG*).
3. Check the appropriate option to preserve file paths (folder names). The unzip process now creates the following subdirectories:

   bin: Contains the runtime executable binary code.

   doc: Contains documentation files.

   include: Contains header files to include in your C / C++ project.

   samples: Contains sample programs in various programming languages.

   There is the option to download the software as MSI file, which makes the installation easier.
4. Optionally register your license key using the License Manager.
5. Identify which interface you are using. Perform the specific installation steps for that interface described in chapter Interfaces.

2.2 Unix

This section describes installation steps required on all Unix platforms, which includes Linux, Mac OS X, Sun Solaris, IBM AIX, HP-UX, FreeBSD and others.

The Unix version of the 3-Heights™ PDF Merge Split API provides two interfaces:

- Java interface
- Native C interface

Here is an overview of the shared libraries and other files that come with the 3-Heights™ PDF Merge Split API:

Table: File Description	
Name	**Description**
`bin/libPdfSplMrgAPI.so`	This is the shared library that contains the main functionality. The file extension varies depending on the UNIX platform.
`doc/*.*`	Documentation
`bin/MSPA.jar`	Java API archive.
`include/*.h`	Contains files to include in your C/C++ Project.

Example code written in different programming languages are available at product page of the PDF Tools AG website (http://www.pdf-tools.com).

All Unix Platforms

1. Unpack the archive in an installation directory, e.g. */opt/pdf-tools.com/*
2. Copy or link the shared object into one of the standard library directories, e.g:
   `ln -s /opt/pdf-tools.com/bin/libPdfSplMrgAPI.so /usr/lib`
3. Verify that the GNU shared libraries required by the product are available on your system:
   - *On Linux*: `ldd libPdfSplMrgAPI.so`
   - *On AIX*: `dump -H libPdfSplMrgAPI.so`

   In case you have not installed the GNU shared libraries yet, proceed as follows:
   (a) Go to http://www.pdf-tools.com and navigate to "Support" → "Resouces".
   (b) Download the GNU shared libraries for your platform.
   (c) Extract the archive and copy or link the libraries into your library directory, e.g */usr/lib* or */usr/lib64*.
   (d) Verify that the GNU shared libraries required by the product are available on your system now.
4. Optionally register your license key using the Command Line License Manager Tool.
5. Identify which interface you are using. Perform the specific installation steps for that interface described in chapter Interfaces.

MAC OS/X

The shared library must have the extension .jnilib for use with Java. We suggest that you create a file link for this purpose by using the following command:

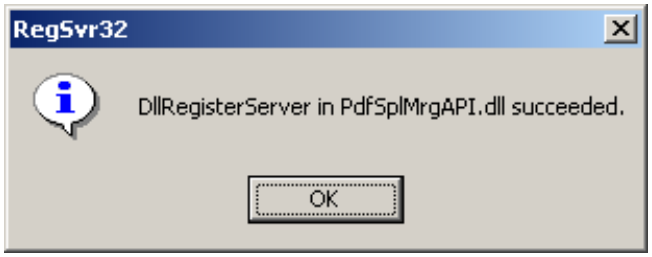
```
ln libPdfSplMrgAPI.dylib libPdfSplMrgAPI.jnilib
```

COM Interface

Before you can use the 3-Heights™ PDF Merge Split API component in your COM application program you have to register the component using the `regsvr32.exe` program that is provided with the Windows operating system; it resides in the directory System32. If you are using a newer operating system, such as Vista or Windows 7, start the command prompt as Administrator. The following screenshot shows the registration of the PDF Merge Split API DLL.

If you are using a 64-bit operating system and would like to register the 32-bit version of the 3-Heights™ PDF Merge Split API, you need to use the regsvr32 from the directory `%SystemRoot%\SysWOW64` instead of `%SystemRoot%\System32`.[1]

If the registration process succeeds, the following dialog window is displayed:



The registration can also be done silently (e.g. for deployment) using the switch `/s`.

The other DLLs do not need to be registered, but for simplicity it is suggested that they are in the same directory as the *PdfSplMrgAPI.dll*.

Java Interface

For compilation and execution: When using the Java interface, the Java-wrapper *jar\MSPA.jar* needs to be on the CLASSPATH. This can be done by either adding it to the environment variable CLASSPATH, or by specifying it using the switch -classpath

```
javac -classpath .;C:\pdf-tools\jar\MSPA.jar sample.java
```

For execution: Additionally the library *PdfSplMrgAPI.dll* needs to be on the Java system property `java.library.path`. This can be achieved by either adding it dynamically at program startup before using the API, or by specifying it using the switch `-Djava.library.path` when starting the Java VM.

```
java -classpath .;C:\pdf-tools\jar\MSPA.jar -Djava.library.path=.;C:\pdf-tools\bin
    sample
```

[1] Otherwise you get the following message: *"LoadLibrary("PdfSplMrgAPI.dll") failed - The specified module could not be found."*

.NET Interface

The 3-Heights™ PDF Merge Split API does not provide a pure .NET solution. Instead, it consists of .NET assemblies, which are added to the project and a native DLL, which is called by the .NET assemblies. This has to be accounted for when installing and deploying the tool.

The .NET assemblies (**NET.dll*) are to be added as references to the project. They are required at compilation time. See also chapter “Getting Started”.

PdfSplMrgAPI.dll is not a .NET assembly, but a native DLL. It is not to be added as a reference in the project.

The native DLL *PdfSplMrgAPI.dll* is called by the .NET assembly *PdfSplMrgNET.dll*.

PdfSplMrgAPI.dll must be found at execution time by the Windows operating system. The common way to do this is adding *PdfSplMrgAPI.dll* as an existing item to the project and set its property “Copy to output directory” to “Copy if newer”.

Alternatively the directory where *PdfSplMrgAPI.dll* resides can be added to the environment variable “PATH” or it can simply be copied manually to the output directory.

Native C Interface

The header file *PdfSplMrgapi_c.h* needs to be included in the C/C++ program.

The library *PdfSplMrgAPI.lib* needs to be linked to the project.

2.3 Uninstall, Install a New Version

If you used the MSI for the installation, go to Start ->3-Heights™ PDF Merge Split API ... ->Uninstall...
If you used the ZIP file: In order to uninstall the product undo all the steps done during installation, e.g. un-register using regsvr32 -u, delete all files, etc.

Installing a new version does not require to previously uninstall the old version. The files of the old version can directly be overwritten with the new version. If using the COM interface, the new DLL must be registered, un-registering the old version is not required.

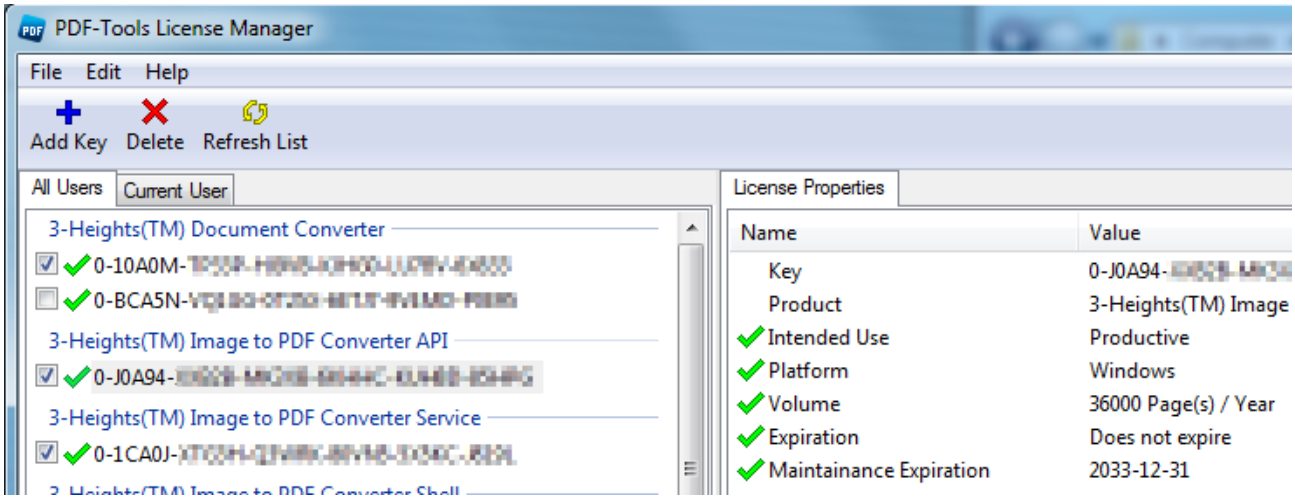
3 License Management

There are three possibilities to pass the license key to the application:

1. The license key is installed using the GUI tool (Graphical user interface). This is the easiest way if the licenses are managed manually. It is only available on Windows.
2. The license key is installed using the shell tool. This is the preferred solution for all non-Windows systems and for automated license management.
3. The license key is passed to the application at runtime via the “LicenseKey” property. This is the preferred solution for OEM scenarios.

3.1 Graphical License Manager Tool

The GUI tool *LicenseManager.exe* is located in the *bin* directory of the product kit.



List all installed license keys

The license manager always shows a list of all installed license keys in the left pane of the window. This includes licenses of other PDF Tools products. The user can choose between:

- Licenses available for all users. Administrator rights are needed for modifications.
- Licenses available for the current user only.

Add and delete license keys

License keys can be added or deleted with the “Add Key” and “Delete” buttons in the toolbar.

- The “Add key” button installs the license key into the currently selected list.
- The “Delete” button deletes the currently selected license keys.

Display the properties of a license

If a license is selected in the license list, its properties are displayed in the right pane of the window.

Select between different license keys for a single product

More than one license key can be installed for a specific product. The checkbox on the left side in the license list marks the currently active license key.

3.2 Command Line License Manager Tool

The command line license manager tool `licmgr` is available in the bin directory for all platforms except Windows.

A complete description of all commands and options can be obtained by running the program without parameters:

```
licmgr
```

List all installed license keys

```
licmgr list
```

Add and delete license keys

Install new license key:

```
licmgr store X-XXXXX-XXXXX-XXXXX-XXXXX-XXXXX-XXXXX
```

Delete old license key:

```
licmgr delete X-XXXXX-XXXXX-XXXXX-XXXXX-XXXXX-XXXXX
```

Both commands have the optional argument `-s` that defines the scope of the action:

- `g`: For all users
- `u`: Current user

Select between different license keys for a single product

```
licmgr select X-XXXXX-XXXXX-XXXXX-XXXXX-XXXXX-XXXXX
```

3.3 License Key Storage

Depending on the platform the license management system uses different stores for the license keys.

Windows

The license keys are stored in the registry:

- `HKLM\Software\PDF Tools AG` (for all users)
- `HKCU\Software\PDF Tools AG` (for the current user)

Mac OS X

The license keys are stored in the file system:

- `/Library/Application Support/PDF Tools AG` (for all users)
- `~/Library/Application Support/PDF Tools AG` (for the current user)

Unix/Linux

The license keys are stored in the file system:

- `/etc/opt/pdf-tools` (for all users)
- `~/.pdf-tools` (for the current user)

Note: The user, group and permissions of those directories are set explicitly by the license manager tool. It may be necessary to change permissions to make the licenses readable for all users. Example:

```
chmod -R go+rx /etc/opt/pdf-tools
```

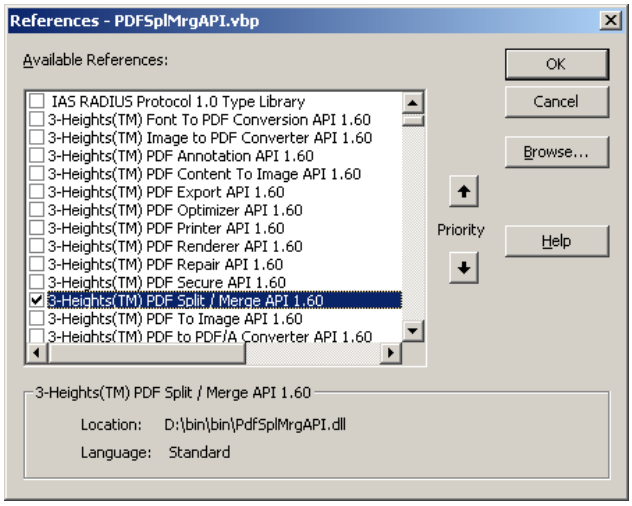
4 Programming Interfaces

4.1 Visual Basic 6

After installing the 3-Heights™ PDF Merge Split API and registering the COM interface (see chapter Download and Installation), you find a Visual Basic 6 example *PdfSplMrgAPI.vbp* in the directory *samples/VB/*. You can either use this sample as a base for an application, or you can start from scratch.

If you start from scratch, here is a quick start guide for you:

1. First create a new Standard-Exe Visual Basic 6 project. Then include the 3-Heights™ PDF Merge Split API component to your project.


References - PDFSplMrgAPI.vbp
Available References:
OK
Cancel
Browse...
Priority
Help
IAS RADIUS Protocol 1.0 Type Library
3-Heights(TM) Font To PDF Conversion API 1.60
3-Heights(TM) Image to PDF Converter API 1.60
3-Heights(TM) PDF Annotation API 1.60
3-Heights(TM) PDF Content To Image API 1.60
3-Heights(TM) PDF Export API 1.60
3-Heights(TM) PDF Optimizer API 1.60
3-Heights(TM) PDF Printer API 1.60
3-Heights(TM) PDF Renderer API 1.60
3-Heights(TM) PDF Repair API 1.60
3-Heights(TM) PDF Secure API 1.60
3-Heights(TM) PDF Split / Merge API 1.60
3-Heights(TM) PDF To Image API 1.60
3-Heights(TM) PDF to PDF/A Converter API 1.60
3-Heights(TM) PDF Split / Merge API 1.60
Location: D:\bin\bin\PdfSplMrgAPI.dll
Language: Standard

2. Draw a new Command Button and optionally rename it if you like.
3. Double-click the command button and insert the few lines of code below. All that you need to change is the path of the file name.

Example:

```
Private Sub Command1_Click()
  Dim InDoc As New PDFSPLMRGAPILib.InDoc
  Dim OutDoc1 As New PDFSPLMRGAPILib.OutDoc
  Dim OutDoc2 As New PDFSPLMRGAPILib.OutDoc

  'Create 2 output files, one of them being encrypted, printing allowed
  OutDoc1.Create "P:\PDF\out1.pdf", "", "owner", ePermPrint
  OutDoc2.Create "P:\PDF\out2.pdf"
  InDoc.Open "P:\PDF\input.pdf"
  OutDoc1.CopyPages InDoc, 1, 2
  OutDoc2.CopyPages InDoc, 3, -1

  InDoc.Close
  OutDoc1.Close
  OutDoc2.Close
End Sub
```

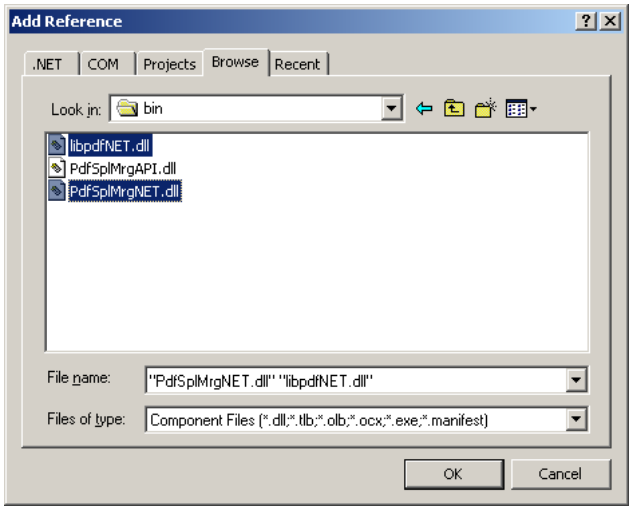
4.2 .NET

As opposed to previous versions, the Windows build numbers 1.7.1.* and later provide a .NET interface.

There should be at least one .NET sample for MS Visual Studio 2005 available in the ZIP archive of the Windows Version of the 3-Heights™ PDF Merge Split API. The easiest for a quick start is to refer to this sample.

In order to create a new project from scratch, do the following steps:

1. Start Visual Studio and create a new C# or VB project.
2. Add a reference to the .NET assemblies. To do so, in the "Solution Explorer" right-click your project and select "Add Reference...". The "Add Reference" dialog will appear. In the tab "Browse", browse for the .NET assemblies *libpdfNET.dll* and *PdfSplMrgNET.dll*
   Add them to the project as shown below:

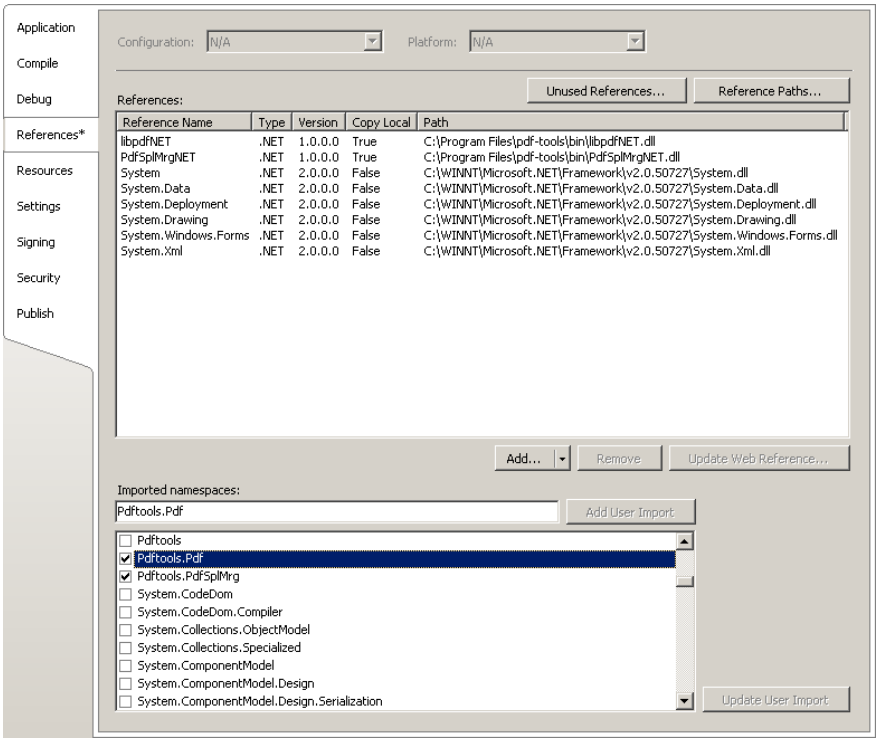


3. import namespaces (Note: This step is optional, but useful.)
4. Write Code

Steps 3 and 4 are shown separately for C# and Visual Basic.

Visual Basic

3. Double-click “My Project” to view its properties. On the left hand side, select the menu “References”. The .NET assemblies you added before should show up in the upper window.
   In the lower window import the namespaces *Pdftools.Pdf* and *Pdftools.PdfSpMrg.*
   You should now have settings similar as in the screenshot below:



4. The .NET interface can now be used as shown below:

```
Dim indoc As New Pdftools.PdfSplMrg.InDoc
Dim outdoc As New Pdftools.PdfSplMrg.OutDoc
indoc.Open(...)
...
```

C#

3. Add the following namespaces:

```
using Pdftools.Pdf;
using Pdftools.PdfSplMrg;
```

4. The .NET interface can now be used as shown below:

```
InDoc indoc = new InDoc();
OutDoc outdoc = new OutDoc();
indoc.Open(...);
...
```

Deploying in .NET

When deploying a .NET solution, please refer to the following FAQ “Deploying in .NET”:
http://www.pdf-tools.com/pdf/Support/FAQ/Article.aspx?name=Deployment-In-NET

Troubleshooting: TypeInitializationException

The most common issue when using the .NET interface is if the native DLL is not found at execution time. This normally manifests when the constructor is called for the first time and exception is thrown - normally of type System.TypeInitializationException.

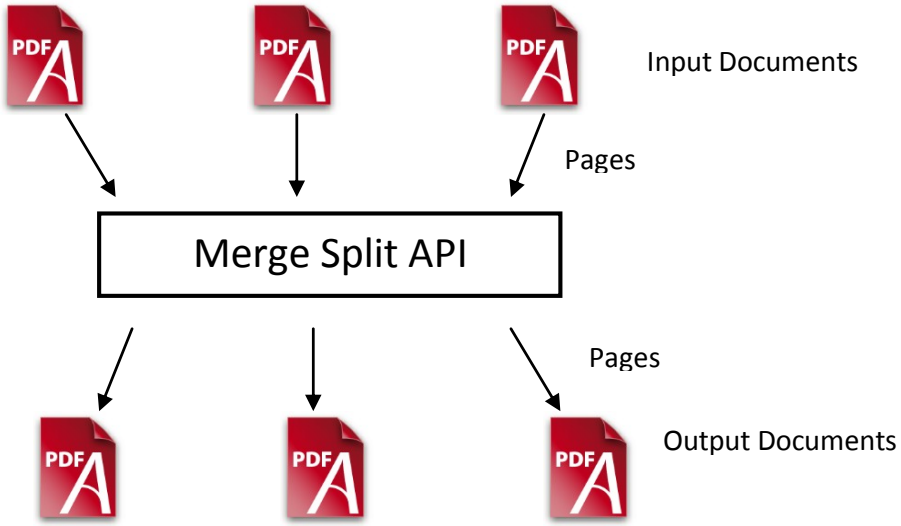
To resolve that ensure the native DLL is found at execution time. For this, see section .NET Interface in the chapter Installation or the following FAQ:
https://www.pdf-tools.com/pdf/Support/FAQ/Article.aspx?name=Exception-type-initializer

5 User's Guide

5.1 Basics

The 3-Heights™ PDF Merge Split API uses a multiple-in/multiple-out architecture. This means it can keep multiple inputs and outputs open and copy pages from the input to the output documents.

This allows for efficient merge and split operations.



Input and output documents can be files or streams.

The 3-Heights™ PDF Merge Split API not only merges and splits pages, but also resources (images, fonts, color spaces, etc.), form fields and outlines (bookmarks). That means if a large document is split into several smaller documents, and they are then merged back into one document, it should result in the original document.

5.2 How to Create PDF/A Compliant Documents

The 3-Heights™ PDF Merge Split API does not have a built-in PDF to PDF/A Converter. This means it can only create PDF/A compliant documents if the input-documents were PDF/A compliant already.

Items that need to be considered when creating PDF/A documents are:

1. Output Intent: Every PDF/A document that uses device specific color spaces (which is the case for most files, even if they only contain text in black color) must have an output intent embedded.
   - The output intent can either be copied from an input file using CopyOutputIntent,
   - or it can be taken from a color profile that resides on the operating system using OutputIntent.

   In either case it is required that the output intent is set before any pages are copied from input to output.
2. Metadata: A PDF/A document requires to have XML Metadata. There are two ways to set the metadata.
   - Copy the from an existing input document using CopyMetadata.
   - Set them directly using SetXMPMetadata or SetXMPMetadataMem.
3. Encryption: PDF/A does not allow encryption. In order to prevent encrypting a PDF output document, make sure to:

- set owner and user password to an empty string, and
- set the permissions flags to ePermNoEncryption

Code snippet in C#:

```
outdoc.Create("output.pdf", "", "", PDFPermission.ePermNoEncryption);
```

4. Tagging information: PDF/A level A conformance requires the document to be tagged. Make sure to copy tagging information by setting the CopyLogicalStructure option to True.
5. Embedded files and Associated Files: Embedded files can be copied using CopyEmbeddedFiles. For PDF/A-3 compliance, the property CopyAssociatedFiles must be set to True in order to copy the embedded file's associations as well.

5.3 How to Efficiently Use the API

How to Copy Pages

The Merge Split API supports copying single pages and page ranges. Have a look at the following two VB code snippets:

Example: Efficient use of API.

```
Dim outdoc As New PDFSPLMRGAPILib.OutDoc
Dim indoc As New PDFSPLMRGAPILib.InDoc
outdoc.Create(...)
indoc.Open(...)
outdoc.CopyPages indoc, 1, 10
indoc.Close()
outdoc.Close()
```

Example: Inefficient use of API.

```
Dim outdoc As New PDFSPLMRGAPILib.OutDoc
Dim indoc As New PDFSPLMRGAPILib.InDoc
outdoc.Create(...)
indoc.Open(...)
For i = 1 to 10
  outdoc.CopyPages indoc, i, i
Next i
indoc.Close()
outdoc.Close()
```

The resulting output file is the same. However the first method is much more efficient. Why? Because when copying a page, it's not just the page content that needs to be copied but also data structures in the document's catalogue dictionary. Examples are outlines, form fields or named destinations. These data structures have to be processed as a whole for each page range. This happens 1 time in the first and 10 times in the second code snippet.

If performance is crucial and for some reason you have to copy multiple times from the same input to the same output document, you can improve performance by:

- Set option `CopyOutlines` to `false`.
- Set option `RemoveNamedDests` to `true`.

Operation on Multiple Documents

While working with multiple input and or output documents special care should be taken to keep at any time as few documents open as possible. The Close methods of the input and output documents should be called at the earliest time possible in order to free associated resources.

Features and their Impact on Performance

The performance of the PDF Merge Split API is determined by the complexity of the input documents as well as by the options chosen. The following options have an impact on performance (ordered by effect on performance):

- Tagged PDF: (Option CopyLogicalStructure) Copying and merging of logical structure information is complex and requires both time and memory. You may deactivate this option, if performance is more important to you than preserving tagging information.
- Interactive Form Fields: (Option CopyForms) If the input documents contain interactive form fields that need not be editable in the output document, the FlattenFormFields option can be used. This speeds the merge process up significantly while preserving the visual appearance of form fields.
- Optimize Resources: (Option OptimizeResources) Detecting and merging duplicate resources takes both time and memory.
- Outlines: (Option CopyOutlines) Copying and merging of outlines takes some time.
- Named Destinations: (Option RemoveNamedDests) Copying and merging of named destinations takes both time and memory.

6 Programmer's Reference

Note this manual describes the COM interface only. Other interfaces (C, Java, .NET) however work similarly, i.e. they have calls with similar names and the call sequence to be used is the same as with COM.

6.1 The InDoc Interface

Close

```
Method Boolean Close()
```

Close an opened input file. If the document is already closed the method does nothing.

Return value:
- True: The file was closed successfully.
- False: Otherwise

CropBox

```
Property Variant CropBox
Accessors: Get
```

This property returns the crop box of the current page defined by the property `Page`. The crop box rectangle is described by the coordinates left, bottom, right, top. If no crop box is defined, the media box is returned. The values are returned as an array of four single precision real numbers. This property cannot be set.

In order to get the dimensions of the page as it is displayed by a viewer application, the rectangle must be rotated according to the property `Rotate`.

ErrorCode

```
Property TPDFErrorCode ErrorCode
Accessors: Get
```

This property can be accessed to receive the latest error code. See also enumeration TPDFErrorCode. PDFTools error codes are listed in the header file *pdferror.h*. Please note, that only few of them are relevant for the 3-Heights™ PDF Merge Split API.

ErrorMessage

```
Property String ErrorMessage
Accessors: Get
```

Return the error message text associated with the last error (see property ErrorCode).

Note, that the property is NULL, if no message is available.

GetXMPMetadataMem

```
Method Variant GetXMPMetadataMem()
```

Returns XMP metadata as byte-array.

GetXMPMetadata

```
Method Boolean GetXMPMetadata(String FileName)
```

Write the XMP metadata to the specified file.

InfoEntry

```
Property String InfoEntry(String Key)
Accessors: Get, Set
```

Parameters:
- Key: A key as string

Return value:
- Value: The value as string

Retrieve or add a key-value pair to the documents info object. Values of predefined keys are also stored in the XMP metadata package.

Popular entries defined by the PDF Specification and used by most PDF viewers are “Title”, “Author”, “Subject”, “Creator” (sometimes referred to as Application) and “Producer” (sometimes referred to as PDF Creator).

Examples in Visual Basic 6:

- Get document title: `t = InfoEntry("Title")`
- Set document title: `InfoEntry( "Title ") = "my title"`
- Set the creation date to 13:55:33, April 5, 2010, UTC+2:
  `InfoEntry("CreationDate") = "D: 20100405135533 + 02'00'"`

MediaBox

```
Property Variant MediaBox
Accessors: Get
```

This property returns the media box of the current page defined by the property `Page`. The media box rectangle is described by the coordinates left, bottom, right, top. The values are returned as an array of four single precision real numbers. The media box is required, it defines the physical boundaries of the medium on which the page is intended to be displayed or printed.

Usually the `MediaBox` is rotated by viewer applications according to the property `Rotate`.

Open

```
Method Boolean Open(String FileName, String Password)
```

Open a PDF file or raster image file, i.e. make the objects contained in the document accessible. If a document is already open, it is closed first.

Parameters:
- FileName: The file name and optionally the file path, drive or server string according to the operating systems file name specification rules.
- Password (optional): The user or the owner password of the encrypted PDF document. If this parameter is left out an empty string is used as a default.

Return value:
- True: The file could successfully be opened.
- False: The file does not exist, it is corrupt, or the password is not valid. Use the property ErrorCode for additional information.

OpenMem

```
Method Boolean OpenMem(Variant MemBlock, String Password)
```

Open a PDF document or raster image from memory, i.e. make the objects contained in the document accessible. If a document is already open, it is closed first.

Parameters:
- MemBlock: The memory block containing the PDF file given as a one dimensional byte array.
- Password (optional): The user or the owner password of the encrypted PDF document. If this parameter is left out an empty string is used as a default.

Return value:
- True: The document could successfully be opened.
- False: The document could not be opened, it is corrupt, or the password is not valid.

Page

```
Property Long Page
Accessors: Get, Set
```

This property allows to set and get the currently selected page of an open document given its page number.

The numbers are counted from 1 for the first page to the value of the PageCount attribute for the last page. If the document is closed zero is returned.

PageCount

```
Property Long PageCount
Accessors: Get
```

Return the number of pages of an open document. If the document is closed then 0 is returned.

Rotate

```
Property Integer Rotate
Accessors: Get
```

Return the rotation value of the page. This value is used by viewer programs to turn the page by the given number of degrees while displaying. A positive number turns the page clockwise. The value must be a multiple of 90, i.e. valid values are -270, -180, -90, 0, 90, 180, 270.

6.2 The OutDoc Interface

AddAssociatedFile

```
Method Boolean AddAssociatedFile(String FileName, String Name, Integer Associate,
String AFRelationship, String MimeType, String Description, DATE ModDate)
```

Add a file to the document's embedded files. For PDF/A-3, the embedded file is associated with an object of the document, i.e. it is an associated file.

This method must be called after Create and before Close. The file is embedded as-is. Embedding files is not allowed for PDF/A-1 and restricted to PDF/A compliant files for PDF/A-2.

Parameters:

- FileName: The path (or URL) to the file to be embedded.
- Name: The name used for the embedded file. This name is presented to the user when viewing the list of embedded files. Default: FileName with the path removed
- Associate: The object to associate the embedded file with. -1 for none, 0 for document, number greater than 0 for respective page. Default: -1. If the embedded file is associated with a page, the page must have been copied already.
- AFRelationship: The relationship of the embedded file to the object associate. (Ignored, if Associate is -1). Allowed values are "Source", "Data", "Alternative", "Supplement", and "Unspecified". Default: "Unspecified".
- MimeType: mime-type of the embedded file. Default: "application/octet-stream". Other common values are "application/pdf", "application/xml", or "application/msword".
- Description: A description of the embedded file. This is presented to the user when viewing the list of embedded files.
- ModDate: The modify date of the file. Default: The modify date of the file on the file system or current time, if not available.

Return value:

- True: The file was embedded successfully.
- False: otherwise.

Example

```
outdoc.AddAssociatedFile "c:\data\input.doc", "", 0, "Source",
    "application/msword", "", 0
```

AddEmbeddedFile

```
Method Boolean AddEmbeddedFile(String FileName, String Name)
```

This is a simplified call that is equal to AddAssociatedFile with default arguments. This is for convenience, for example when embedding files in a PDF/A-2 conforming document.

AddOutlineItem

```
Method Boolean AddOutlineItem(String Text, Long PageNo, Single Left, Single Bottom,
Single Right, Single Top, Single Zoom, Integer Mode)
```

This method adds a new outline (bookmark) item including name and its position. To apply multiple outlines, call the function multiple times. The relevant parameters depend on the parameter Mode.

The outline item is inserted at the end of the existing outline tree.

Parameters:

- Text: the displayed text.
- PageNo: the target page number in this document.
- Left: the left position in points.
- Bottom: the bottom position in points.
- Right: the right position in points.
- Top: the top position in points.
- Zoom: the zoom level; 1 = 100%.
- Mode: the destination type.
  - 0: /XYZ left top zoom
  - 1: /Fit
  - 2: /FitH top
  - 3: /FitV left
  - 4: /FitR left bottom right top
  - 5: /FitB
  - 6: /FitBH top
  - 7: /FitBW left

  The destination types are described in the chapter “Destinations” of the PDF Reference (table “Destination Syntax”).

Example

```
Dim outdoc As New PDFSPLMRGAPILib.outdoc
Dim indoc As New PDFSPLMRGAPILib.indoc
If indoc.Open("in.pdf", "") Then
  outdoc.Create "out.pdf"
  outdoc.CopyOutlines = False
  outdoc.CopyPages indoc
  outdoc.AddOutlineItem "Page 1, Fit", 1, 0, 0, 0, 0, 0, 1
  outdoc.AddOutlineItem "Page 2, @50,400,300%", 2, 50, 0, 0, 400, 3, 0
  outdoc.AddOutlineItem "Page 3, FitH", 3, 0, 0, 0, 2000, 0, 2
  outdoc.Close
  indoc.Close
End If
Set indoc = Nothing
Set outdoc = Nothing
```

AddOutlineItem2

```
Method Boolean AddOutlineItem2(String Text, Long PageNo, Single Left, Single
    Bottom,
Single Right, Single Top, Single Zoom, Integer Mode, Integer Level, Boolean Open)
```

This method adds a new outline (bookmark) item including name and its position. To apply multiple outlines, call the function multiple times. The relevant parameters depend on the parameter Mode.

The outline item is inserted in the end of the existing outline tree at the hierarchy level specified. This method can be used to create an arbitrary outline hierarchy and can be used in combination with the method CopyOutlineItems.

Parameters:
- Text: the displayed text.
- PageNo: the target page number in this document.
- Left: the left position in points.
- Bottom: the bottom position in points.
- Right: the right position in points.
- Top: the top position in points.
- Zoom: the zoom level; 1 = 100%.
- Mode: the destination type.
  - 0: /XYZ left top zoom
  - 1: /Fit
  - 2: /FitH top
  - 3: /FitV left
  - 4: /FitR left bottom right top
  - 5: /FitB
  - 6: /FitBH top
  - 7: /FitBW left

  The destination types are described in the chapter “Destinations” of the PDF Reference (table “Destination Syntax”).
- Level: The hierarchy level within the existing outline tree at which the outline item is appended. Level 0 for top level.
- Open: Whether or not this outline should initially be open (child outline items expanded) or closed (child outline items collapsed).

Author

```
Property String Author
Accessors: Get, Set
Default: ""
```

This property sets the Author attribute in the document.

Close

```
Method Boolean Close()
```

Close an opened input file. If the document is already closed the method does nothing.

Return value:
- True: The file was closed successfully.
- False: Otherwise

CopyAssociatedFiles

```
Property Boolean CopyAssociatedFiles
Accessors: Get, Set
Default: false
```

If this option is set to true, associated files are copied in CopyPages and CopyEmbeddedFiles. For PDF/A-3 compliance, this property must be set to true and the method CopyEmbeddedFiles must be called.

CopyEmbeddedFiles

```
Method Boolean CopyEmbeddedFiles(InDoc InDoc, Boolean All)
```

This method copies embedded files from the input document. If the option CopyAssociatedFiles is set, the document’s associated files are copied as well.

Parameters:
- InDoc: the input document.
- All: If set to true: Copy all embedded files. If set to false: Copy embedded files associated with document and pages copied with CopyPages only (PDF/A-3 only, CopyAssociatedFiles must be set to true)

CopyForms

```
Property Boolean CopyForms
Accessors: Get, Set
Default: true
```

Get the option to copy interactive form fields in CopyPages.

CopyLogicalStructure

```
Property Boolean CopyLogicalStructure
Accessors: Get, Set
Default: true
```

Get the option to copy logical structure information in CopyPages. It is recommended to set this option, if all input documents are Tagged PDF. Otherwise this could be deactivated in order to create smaller output files and get a much better performance of CopyPages. This option is required for PDF/A level A conformance.

CopyMetadata

```
Method Boolean CopyMetadata(InDoc InDoc)
```

Copy info object and XMP metadata from an InDoc. This method should only be called once per output document.

Setting the XMP metadata automatically adjusts and thereby overrides the current document info entries. Therefore document info entries must always be applied after setting the XMP Metadata to become effective (applies to properties Author, Keywords, Subject and Title).

CopyOptionalContent

```
Method Boolean CopyOptionalContent(InDoc InDoc, String Text)
```

Copy configuration data for optional content groups (layers). The groups are collected under the name given in the parameter “Text”. If this parameter is NULL, then the document title of InDoc is used as “Text”, unless the document title is empty in which case the file name is chosen.

This method is intended to be used prior to calling CopyPages when MergeOptionalContent is set to False.

CopyOutlines

```
Property Boolean CopyOutlines
Accessors: Get, Set
Default: true
```

Get the option to copy outlines in CopyPages.

CopyOutlineItems

```
Method Boolean CopyOutlineItems(InDoc InDoc, Long FirstPage, Long LastPage,
    Integer Level)
```

Copy outline items of the page range. The outline items are inserted in the end of the existing outline tree at the hierarchy level specified. The pages of the page range must be copied using CopyPages before calling the CopyOutlineItems method. It is recommended to set the CopyOutlines option to False.

The method CopyOutlineItems can be used in combination with the AddOutlineItem method.

Parameters:

- InDoc: the input document.
- FirstPage, LastPage: the page range of the input document. All outline items belonging to this page range are copied.
- Level: The hierarchy level within the existing outline tree at which the outline items are appended. Level 0 for top level.

Example

```
Dim outdoc As New PDFSPLMRGAPILib.outdoc
Dim indoc As New PDFSPLMRGAPILib.indoc
If indoc.Open("in.pdf", "") Then
  outdoc.Create "out.pdf"
  outdoc.CopyOutlines = False
  outdoc.CopyPages indoc
  outdoc.AddOutlineItem2 "in.pdf ", 1, 0, 0, 0, 0, 0, 1, 0, True
  outdoc.CopyOutlineItems indoc, 1, -1, 1
  outdoc.Close
  indoc.Close
End If
Set indoc = Nothing
Set outdoc = Nothing
```

CopyOutputIntent

```
Method Boolean CopyOutputIntent(InDoc InDoc)
```

Copy the PDF/A output intent. This method should only be called once per output document. It should be called prior to copying any pages.

CopyPages

```
Method Boolean CopyPages(InDoc InDoc, Long FirstPage, Long LastPage)
```

This method copies a range of pages from the InDoc. The method returns true if all pages were copied successfully. Depending on the options set, all outlines and form fields associated with the pages are copied as well.

CopyViewerProperties

```
Method Boolean CopyViewerProperties(InDoc InDoc)
```

Copy viewer properties, which include: Page Layout, Page Mode, Open Actions, Piece Info and Collection properties (if CopyEmbeddedFiles is used only).

Create

```
Method Boolean Create(String FileName, String UserPassword, String OwnerPassword,
TPDFPermission PermissionFlags)
Method Boolean Create2(String FileName, String UserPassword, String OwnerPassword,
TPDFPermission PermissionFlags, Long KeyLength, String StrF, String StmF)
```

Create an output PDF document, apply the security settings and save the content from the input file to the output file.

Notes:

- With Version 4.1.13.0 Create2 was added with three new parameters for key length, string filter and stream filter to support AES-V2 and AES-V3 encryption.
- The last three parameters (KeyLength, StrF, StmF) are only relevant in specific cryptographic situations. In all other cases, it is easiest to use the default values 128, "V2", "V2".

Parameters:
- FileName: the file name and optionally the file path, drive or server string according to the operating systems file name specification rules.
- UserPassword (optional): the user password of the encrypted PDF document.
- OwnerPassword (optional): the owner password of the encrypted PDF document. If the owner password is empty, the user password is used instead.
- PermissionFlags (optional): Set the permission flags of the PDF document. This option requires an owner password to be set. By default no permissions are granted. The permissions that can be granted are listed at the enumeration TPDFPermission.
  To not encrypt the output document, set PermissionFlags to -1, user and owner password to 0.
  In order to allow high quality printing, flags ePermPrint and ePermDigitalPrint need to be set.
- KeyLength (optional, Default: 128): The key length is a determining factor of the strength of the encrypting algorithm and the amount of time to break the cryptographic system. For RC4 the key length can be any value from 40 to 128 that is a multiple of 8. For AESV2 the key length is automatically set to 128, for AESV3 to 256.
  Notes:
  - Certain PDF viewers only support 40 and 128 bit encryption. Other tools, such as the 3-Heights™ tools also support other encryption key lengths.
  - 256 bit encryption requires Acrobat 9 or later.
- StrF (optional, Default: "V2"): Set the string crypt filter. Supported values are "None", "V2" (default), "RC4", "AESV2" and "AESV3". Setting this value to an empty string or null, means the default filter is used.
  Supported crypt filter:
  - "None": The application does not decrypt data.
  - "V2" or "RC4": (PDF 1.2) The application asks the security handler for the encryption key and implicitly decrypts data using the RC4 algorithm.
  - "AESV2": (PDF 1.6) The application asks the security handler for the encryption key and implicitly decrypts data using the AES-V2 128 bit algorithm.
  - "AESV3": (PDF 1.7) The application asks the security handler for the encryption key and implicitly decrypts data using the AES-V3 256 bit algorithm.
- StmF (optional, Default: "V2"): Set the stream crypt filter. Supported values are "None", "V2", "RC4", "AESV2" and "AESV3". Note that certain viewers require the stream crypt filter to be equal to the string crypt filter, e.g. both must be RC4 or AES. Other tools, such as the 3-Heights™ PDF tools do not have this limitation. Setting this value to an empty string or null, means the default filter is used.

Return value:
- True: The file was created successfully.
- False: The file was not be created. This can be due to permissions or a locked file, or another reason.

CreateInMemory

```
Method Boolean CreateInMemory()
Method Boolean CreateInMemory2(String UserPassword, String OwnerPassword,
TPDFPermission PermissionFlags, Long KeyLength, String StrF, String StmF)
```

This method saves the output PDF in memory as a byte array. (see also method GetPdf)

For a description of the parameters of CreateInMemory2, see method Create2.

Return value:
- True: The PDF document was created successfully.
- False: otherwise.

ErrorCode

```
Property TPDFErrorCode ErrorCode
Accessors: Get
```

This property can be accessed to receive the latest error code. See also enumeration TPDFErrorCode. PDF-Tools error codes are listed in the header file *pdferror.h*. Please note, that only few of them are relevant for the 3-Heights™ PDF Merge Split API.

ErrorMessage

```
Property String ErrorMessage
Accessors: Get
```

Return the error message text associated with the last error (see property ErrorCode).

Note, that the property is NULL, if no message is available.

FlattenFormFields

```
Property Boolean FlattenFormFields
Accessors: Get, Set
Default: false
```

Get and set the option to flatten all form fields CopyPages[2].

FlattenSigAppearance

```
Property Boolean FlattenSigAppearance
Accessors: Get, Set
Default: false
```

Get and set the option to flatten appearances of signed signature fields CopyPages.

A digital signature consists of two parts: First, a cryptographic part that includes a hash value based on the content of the document that is being signed. If the document is modified at a later time, the computed hash value is no longer correct and the signature becomes invalid, i.e. the validation will fail and will report that the document has been modified since the signature has been applied. Second, an optional visual appearance on a page of the PDF document. The signature appearance can be useful to indicate the presence of a digital signature by a particular signer.

Processing the PDF with 3-Heights™ PDF Merge Split API breaks the signature, and therefore the cryptographic part needs to be removed. In general, the visual appearance is regarded as worthless without the cryptographic part, it is removed by default. The visual appearance can be preserved by setting the property FlattenSigAppearance to true.

GetPDF

```
Method Variant GetPDF()
```

Get the output file from memory. See also method CreateInMemory.

Return value:

- A byte array containing the output PDF. In certain programming languages, such as Visual Basic 6, the type of the byte array must explicitly be Variant.

[2] Up to version 4.5.14.0 of the 3-Heights™ PDF Merge Split API the option `FlattenFormFields` also flattened signed signature appearances. In newer versions the option `FlattenSigAppearance` must be used to get the same result.

InfoEntry

```
Property String InfoEntry(String Key)
Accessors: Get, Set
```

Parameters:
- Key: A key as string

Return value:
- Value: The value as string

Retrieve or add a key-value pair to the documents info object. Values of predefined keys are also stored in the XMP metadata package.

Popular entries defined by the PDF Specification and used by most PDF viewers are “Title”, “Author”, “Subject”, “Creator” (sometimes referred to as Application) and “Producer” (sometimes referred to as PDF Creator).

Examples in Visual Basic 6:

- Get document title: `t = InfoEntry("Title")`
- Set document title: `InfoEntry( "Title ") = "my title"`
- Set the creation date to 13:55:33, April 5, 2010, UTC+2:
  `InfoEntry("CreationDate") = "D: 20100405135533 + 02'00'"`

Keywords

```
Property String Keywords
Accessors: Get, Set
Default: ""
```

This property adds keywords to the document.

LicenseIsValid

```
Property Boolean LicenseIsValid
Accessors: Get
```

Check if the license is valid.

MergeOptionalContent

```
Property Boolean MergeOptionalContent
Accessors: Get, Set
Default: True
```

Merge compatible optional content groups (layers). If True then in the method CopyPages the configuration of optional content is compared with the input file. If it is found to be the same then the optional content groups are assumed to be the same in the input and the output document and merging takes place. If they are different then optional content groups are assumed to be distinct and they are simply added.

If this property is False then no configuration of optional content groups is copied to the output file in CopyPages. In this case one can use the method CopyOptionalContent to copy such configuration information.

OptimizeResources

```
Property Boolean OptimizeResoruces
Accessors: Get, Set
Default: False
```

Find and merge redundant resources from different input files. Equal fonts, images and color spaces are detected. By activating this feature, much smaller output files are created, if similar files are merged. The merging process uses time and memory resources.

OutputIntent

```
Property String OutputIntent
Accessors: Set
Default: ""
```

Load the PDF/A output intent's color profile from specified file.

PageLayout

```
Property TPDFPageLayout PageLayout
Accessors: Set
```

Set the page layout that shall be used when the document is opened. Alternatively the page layout can be copied from an input document using the method CopyViewerProperties.

PageMode

```
Property TPDFPageMode PageMode
Accessors: Set
```

Set the page mode that specifies how the document shall be displayed when opened. Alternatively the page mode can be copied from an input document using the method CopyViewerProperties.

RemoveNamedDests

```
Property Boolean RemoveNamedDests
Accessors: Get, Set
Default: False
```

If this property is set, all named destinations of the input document are removed and all internal named destinations converted to regular destinations.

Use this option with care: If a document is split into multiple documents with the intention of merging the pieces back together at a later time, this option should not be used. The reason is that links in between the pieces will not work after the merge.

Rotate

```
Property Integer Rotate
Accessors: Get, Set
Default: 0
```

The number of degrees by which the page should be rotated additionally when the document is viewed (or printed). The value is applied to the pages at the time they are copied to this output document. The value must be a multiple of 90. A positive value is a clockwise rotation. The default is 0.

SetLicenseKey

```
Method Boolean SetLicenseKey(String Licensekey)
```

Set the license key.

SetOpenAction

```
Method Boolean SetOpenAction(Long PageNo, Single Left, Single Bottom, Single Right,
Single Top, Single Zoom, TPDFDestMode Mode)
```

The open action defines which page shall be presented to the user initially upon opening the document.

Parameters:
- PageNo: the target page number.
- Left: the left position in points.
- Bottom: the bottom position in points.
- Right: the right position in points.
- Top: the top position in points.
- Zoom: the zoom level; 1 = 100%.
- Mode: the destination type. The destination types are described in the chapter “Destinations” of the PDF Reference (table “Destination Syntax”).

SetViewerPreference

```
Method Boolean SetViewerPreference(String Key, String Value)
```

Add an entry to the viewer preferences dictionary. All properties defined in the PDF Reference 1.7 and earlier are supported.

Parameters:
- Key: the name of the entry.
- Value: a string representation of the value. Names: value string, Booleans: “TRUE” or “FALSE”, Integers: decimal numbers like “22”, Arrays: comma-separated list of items like “1, 3, 55”

SetXMPMetadata

```
Method Boolean SetXMPMetadata(String FileName)
```

Load XMP metadata from specified file.

Setting the XMP metadata automatically adjusts and thereby overrides the current document info entries. Therefore document info entries must always be applied after setting the XMP Metadata to become effective (applies to properties Author, Keywords, Subject and Title).

SetXMPMetadataMem

```
Method Boolean SetXMPMetadata(Variant Mem)
```

Load XMP metadata from a byte array. See also SetXMPMetada.

Subject

```
Property String Subject
Accessors: Get, Set
Default: ""
```

This property sets the Subject attribute of the document.

Title

```
Property String Title
Accessors: Get, Set
Default: ""
```

This property sets the Title attribute of the document.

6.3 Enumerations

Note: Depending on the interface, enumerations may have “TPDF” as prefix (COM, C) or “PDF” as prefix (.NET) or no prefix at all (Java).

TPDFErrorCode

All TPDFErrorCode enumerations start with “PDF_” followed by a single letter which is one of “S”, “E”, “W” or “I”, an underscore and a descriptive text. The single letter gives in an indication of the type of error. These are: Success, Error, Warning, Information. With respect to corrupt PDF files: An error indicates a corruption in the PDF, the file may or may not be readable. A warning indicates the file is readable but not valid.

A full list of all PDF Tools error codes is available in the header file *pdferror.h*. Note that only a few are relevant for the PDF Merge Split API. The most common are listed here.

`PDF_S_SUCCESS`	The operation was completed successfully.
`LIC_E_NOTSET`, `LIC_E_NOTFOUND`, …	Various license management related errors.
`PDF_E_FILEOPEN`	Failed to open the file.
`PDF_E_FILECREATE`	Failed to create the file.
`PDF_E_PASSWORD`	The authentication failed due to a wrong password.
`PDF_E_UNKSECHANDLER`	The file uses a proprietary security handler, e.g. for a proprietary digital rights management (DRM) system.
`PDF_E_XFANEEDSRENDERING`	The file contains unrendered XFA form fields, i.e. the file is an XFA and not a PDF file.
`PDF_SPLMRG_W_DOCSIGNED`	Document is signed.
`PDF_SPLMRG_W_RMXFA`	XFA stream was not copied.
`PDF_SPLMRG_W_RMSUBMIT`	SubmitForm action was not copied.
`PDF_SPLMRG_W_PARTSUBMIT`	Partial SubmitForm action altered to submit all fields.
`PDF_SPLMRG_W_RMSIGANNOT`	Signature annotation was not copied.
`PDF_SPLMRG_W_RMVALUE`	Value or default value of field “%field%” was discarded due to field name collision
`PDF_SPLMRG_W_MVFIELD`	Renamed field “%field%” to “%newname%” due to field name collision.

TPDFPermission

An enumeration for permission flags. If a flag is set, the permission is granted.

`ePermNoEncryption`	Do not apply encryption. This enumeration shall not be combined with another enumeration. When using this enumeration set both passwords to an empty string or null.
`ePermAll`	Grant all Permissions
`ePermPrint`	Low resolution printing
`ePermModify`	Changing the document
`ePermCopy`	Content copying or extraction
`ePermAnnotate`	Annotations
`ePermFillForms`	Filling of form fields
`ePermSupportDisabilities`	Support for disabilities
`ePermAssemble`	Document Assembly
`ePermNoEncryption`	No encryption
`ePermDigitalPrint`	High resolution printing

Changing permissions or granting multiple permissions is done using a bitwise or operator. Changing the current permissions in Visual Basic should be done like this:

Allow Printing:	`Permission = Permission Or ePermPrint`
Prohibit Printing:	`Permission = Permission And Not ePermPrint`

7 Examples

Examples in various programming languages are included in the ZIP files of the release and evaluation versions.

8 Licensing and Copyright

The 3-Heights™ PDF Merge Split API is copyrighted. This user's manual is also copyright protected; it may be copied and given away provided that it remains unchanged including the copyright notice.

9 Contact

PDF Tools AG
Kasernenstrasse 1
8184 Bachenbülach
Switzerland
http://www.pdf-tools.com